Transitioning to QuickTime X

Session 314

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Transitioning to QuickTime X

Kevin Calhoun
Tim Monroe
QuickTime Engineering

Introduction

- QuickTime X is a new technology for displaying time-based media
  - Migrated to the desktop from Apple devices
  - Implemented independently of QuickTime 7
  - Provides significant improvements
- QuickTime X is accessible only via QTKit APIs

What You'll Learn

- QuickTime X
  - How to open media using QuickTime X
  - When it makes sense to do so
  - What restrictions apply when doing so
- Asynchronous Loading
  - How to open media without blocking the main thread

Other QTKit Changes in Snow Leopard

- Possible to display one QTMovie object in multiple views or layers

  Deprecated `QTMovieCurrentSizeAttribute`

  - Use `QTMovieNaturalSizeAttribute` to get “authored” size

  Deprecated `QTMovieSizeDidChangeNotification`

  - Use `QTMovieNaturalSizeDidChangeNotification`

QuickTime X

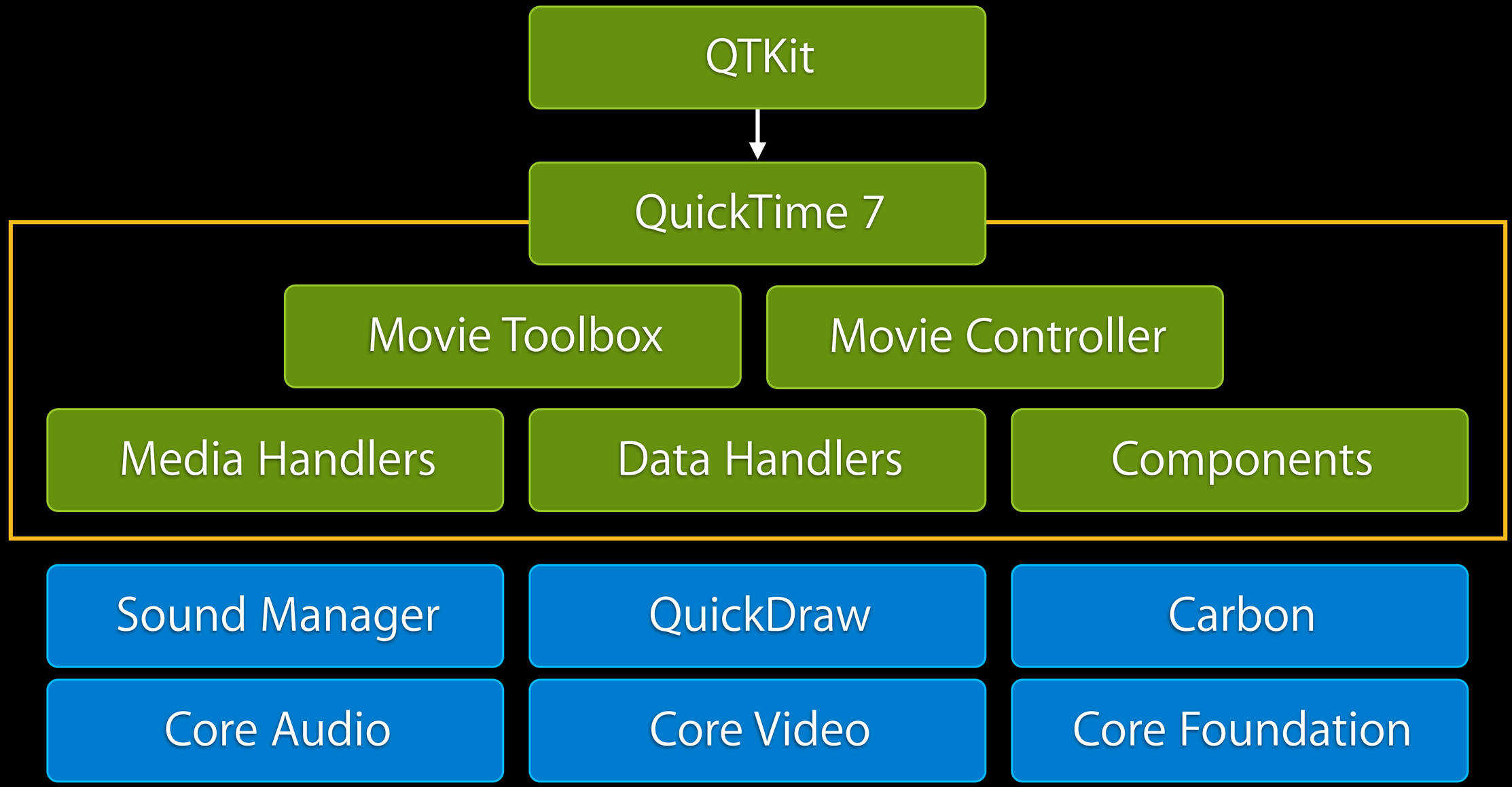
“A QTMovie object represents both a QuickTime movie and a movie controller.”

QTMovie Documentation, Pre-Snow Leopard

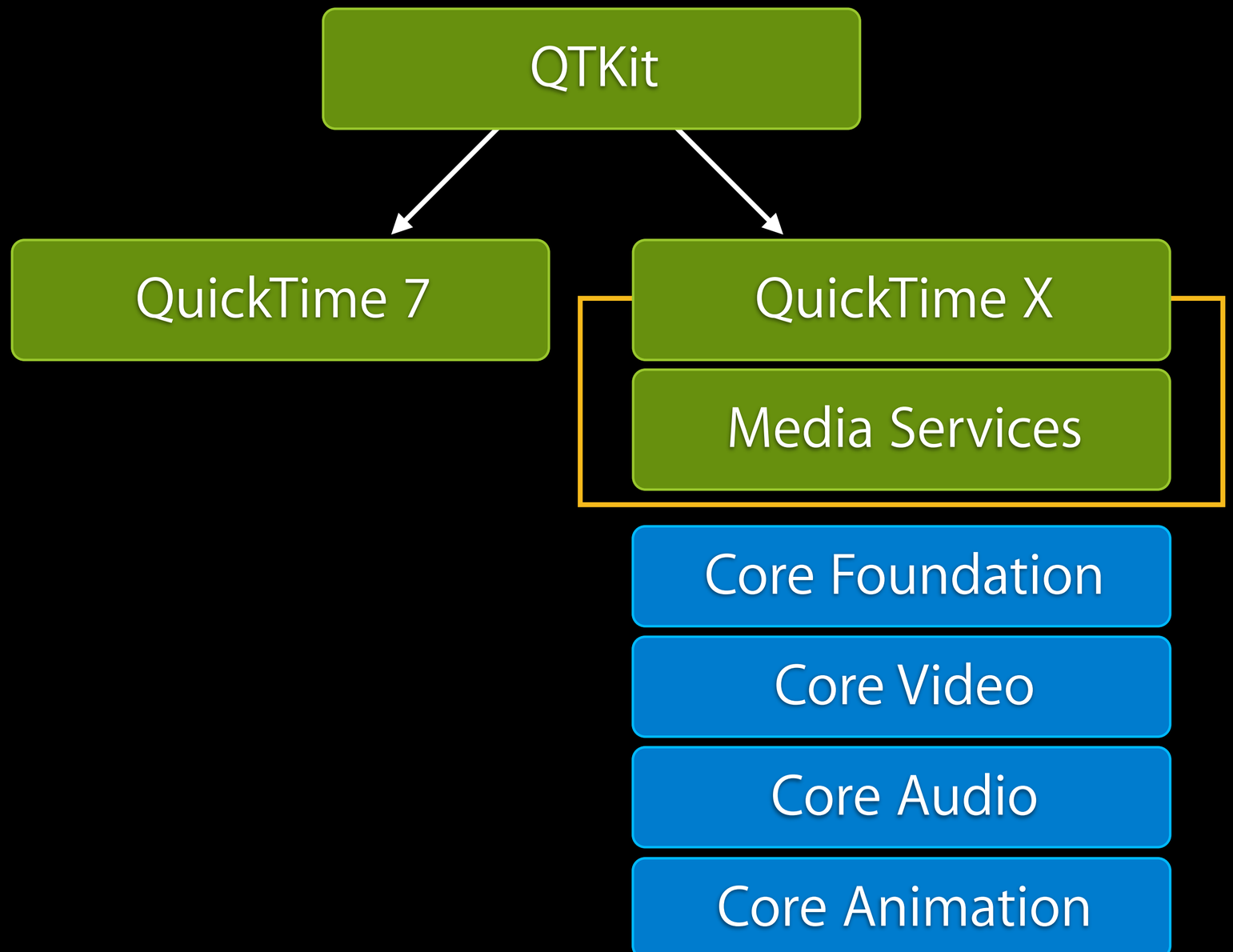
“A QTMovie object represents a playable collection of media data.”

QTMovie Documentation, Snow Leopard and later

QTKit Architecture



QTKit Architecture in Snow Leopard



QuickTime X

- Set of new media services
  - Extremely efficient playback
  - Multithreaded
  - 64-bit native
  - Based on Core Foundation
  - ColorSync throughout
- Handles modern linear media formats
  - Roughly, whatever can be played on Apple devices
- Very simple public API in Snow Leopard

Other Benefits of QuickTime X

- QTKit Capture
  - Now uses QuickTime X to process captured movies
  - Captured movies are color tagged
- QTKit Export
  - Now uses QuickTime X to export to device formats

Opening a Movie File

- Pre Snow Leopard, designated initializer is `-initWithQuickTimeMovie:disposeWhenDone:error:`

```
- (id)initWithQuickTimeMovie:(Movie)movie
        disposeWhenDone:(BOOL)dispose
                  error:(NSError **)errorPtr;
```

Opening a Movie File

- In Snow Leopard, `-initWithAttributes:error:` is now the designated initializer

```
- (id)initWithAttributes:(NSDictionary *)attributes
                   error:(NSError **)errorPtr;

+ (id)movieWithAttributes:(NSDictionary *)attributes
                    error:(NSError **)errorPtr;
```

Data Locator Attributes

```
- (id)initWithAttributes:(NSDictionary *)attributes
                   error:(NSError **)errorPtr;

QTMovieFileNameAttribute
QTMovieURLAttribute
QTMoviePasteboardAttribute
QTMovieDataAttribute
QTMovieDataReferenceAttribute
```

Movie Configuration Attributes

```
- (id)initWithAttributes:(NSDictionary *)attributes
                   error:(NSError **)errorPtr;

QTMovieEditableAttribute
QTMovieDelegateAttribute
QTMovieLoopsAttribute
QTMovieVolumeAttribute
```

Movie Configuration Attributes

```
- (id)initWithAttributes:(NSDictionary *)attributes
                   error:(NSError **)errorPtr;

QTMovieEditableAttribute
QTMovieDelegateAttribute
QTMovieLoopsAttribute
QTMovieVolumeAttribute
QTMovieDurationAttribute?
```

Some Attributes Are Not Settable!

```
/*!
    @constant       QTMovieDurationAttribute
    @abstract       The duration of a QTMovie object; the value f
    @discussion     This attribute can be read but not written.
                    This attribute can be read but not written wh
 */
QTKIT_EXTERN NSString * const QTMovieDurationAttribute
```

Some Attributes Are Not Settable!

```
/*!
    @constant       QTMovieDurationAttribute
    @abstract       The duration of a QTMovie object; the value f
    @discussion     This attribute can be read but not written.
                    This attribute can be read but not written wh
 */
QTKIT_EXTERN NSString * const QTMovieDurationAttribute
```

Special Initialization Attributes

```
- (id)initWithAttributes:(NSDictionary *)attributes
                   error:(NSError **)errorPtr;

QTMovieDontInteractWithUserAttribute
QTMovieResolveDataRefsAttribute
QTMovieAskUnresolvedDataRefsAttribute
QTMovieOpenAsyncOKAttribute
```

Special Initialization Attributes

```
- (id)initWithAttributes:(NSDictionary *)attributes
                   error:(NSError **)errorPtr;

QTMovieOpenForPlaybackAttribute
```

Opening a Movie for Playback

```
QTMovie *movie = nil;
NSError *error = nil;
NSNumber *num = [NSNumber numberWithBool:YES];
NSDictionary *attributes =
                    [NSDictionary dictionaryWithObjectsAndKeys:
                        fileName, QTMovieFileNameAttribute,
                        num, QTMovieLoopsAttribute,
                        num, QTMovieOpenForPlaybackAttribute,
                     nil];

movie = [[QTMovie alloc] initWithAttributes:attributes
                                      error:&error];
```

Opening a Movie for Playback

- Not all media files can be handled by QuickTime X
  - Device-capable media files generally OK
- If not suitable, QTKit falls back to the QuickTime 7 media architecture
- You cannot determine which media architecture is being used
- When requesting open for playback, you are restricted to QTKit APIs that apply to playback only

Some Methods Are Not Callable

```
/*!
    @method         quickTimeMovie
    @abstract       Returns the QuickTime Movie associated with a QTMovie object.
    @discussion     This method cannot be called when the movie has been initialized
                    with QTMovieOpenForPlaybackAttribute set to YES.
                    This method cannot be called by 64-bit applications.
    @result         A QuickTime Movie.
*/
- (Movie)quickTimeMovie;
```

Some Attributes Are Not Settable

```
/*!
    @constant       QTMovieEditableAttribute
    @abstract       The editable setting of a QTMovie object; the value for this key is of type NSNumber
                    This value is YES if the movie can be edited, NO otherwise.
    @discussion     This attribute can be read and written.
                    This attribute can be read but not written
                    when the movie has been initialized with QTMovieOpenForPlaybackAttribute set to YES.
 */
QTKIT_EXTERN NSString * const QTMovieEditableAttribute                  AVAILABLE_QTKIT_VERSION_7_0_A
```

Demos

Playing Movies with QuickTime X

Asynchronous Loading

Asynchronous Loading

- New service provided by QTKit in Snow Leopard
- Do ***everything possible*** to avoid blocking the main thread
  - Try to load the movie on a background thread
  - If that is not possible, fail

Special Initialization Attributes

```
- (id)initWithAttributes:(NSDictionary *)attributes
                   error:(NSError **)errorPtr;

QTMovieOpenAsyncRequiredAttribute
```

Opening a Movie Asynchronously

```
QTMovie *movie = nil;
NSError *error = nil;
NSNumber *num = [NSNumber numberWithBool:YES];
NSDictionary *attributes =
                    [NSDictionary dictionaryWithObjectsAndKeys:
                        fileName, QTMovieFileNameAttribute,
                        num, QTMovieLoopsAttribute,
                        num, QTMovieOpenAsyncRequiredAttribute,
                     nil];

movie = [[QTMovie alloc] initWithAttributes:attributes
                                      error:&error];
```

You Must Monitor Movie Load States

- Relevant movie data may not be available immediately after initializing a QTMovie object
- So, it may not yet be meaningful to query movie attributes or to perform other operations on it

Movie Load States

```
enum {
    QTMovieLoadStateError           = -1L,
    QTMovieLoadStateLoading         = 1000,
    QTMovieLoadStateLoaded          = 2000,
    QTMovieLoadStatePlayable        = 10000,
    QTMovieLoadStatePlaythroughOK   = 20000,
    QTMovieLoadStateComplete        = 100000L
};
typedef NSInteger QTMovieLoadState;
```

Registering for State-Changed Notifications

```
movie = [[QTMovie alloc] initWithAttributes:attributes
                                      error:&error];

[[NSNotificationCenter defaultCenter]
        addObserver:self
           selector:@selector(loadStateChanged:)
               name:QTMovieLoadStateDidChangeNotification
             object:movie];
```

Handling State-Changed Notifications

```
- (void)loadStateChanged:(NSNotification *)notification
{
    QTMovie *movie = [notification object];

    if (movie) {
        [self handleLoadStateChanged:movie];
    }
}
```

Handling State-Changed Notifications

```
- (void)handleLoadStateChanged:(QTMovie *)movie
{
    NSInteger loadState = [[movie
            attributeForKey:QTMovieLoadStateAttribute] longValue];

    if (loadState == QTMovieLoadStateError) {
        /* an error occurred; handle it */
        ...
    }

    if (loadState >= QTMovieLoadStateLoaded) {
        /* can now query movie attributes, if desired */
        [movieView setMovie:movie];
    }
}
```

Registering for State-Changed Notifications

```
movie = [[QTMovie alloc] initWithAttributes:attributes
                                      error:&error];

[self handleLoadStateChanged:movie];

[[NSNotificationCenter defaultCenter]
      addObserver:self
         selector:@selector(loadStateChanged:)
             name:QTMovieLoadStateDidChangeNotification
           object:movie];
```

Asynchronous Loading

- Not all media files can be opened asynchronously
  - For instance, media handler may not be thread-safe
- You ***can*** determine if asynchronous loading failed
- If asynchronous loading fails…
  - …retry without async required
  - …or proceed without the QTMovie object

Detecting Asynchronous Opening Failure

```
- (void)handleLoadStateChanged:(QTMovie *)movie
{
    NSInteger loadState = [[movie
            attributeForKey:QTMovieLoadStateAttribute] longValue];

    if (loadState == QTMovieLoadStateError) {
        NSError *error = [movie attributeForKey:
                                    QTMovieLoadStateErrorAttribute];
        NSString *domain = [error domain];
        NSInteger code = [error code];

        if (/* not file not found or invalid URL ... */) {
            /* retry without QTMovieOpenAsyncRequiredAttribute */
        }
    ...
}
```

Demo

Opening a Movie Asynchronously

You Must Monitor Movie Load States, Except…

- …when you explicitly indicate that a movie must ***not*** be opened asynchronously

```
QTMovie *movie = nil;
NSNumber *async = [NSNumber numberWithBool:NO];
NSDictionary *attributes =
                    [NSDictionary dictionaryWithObjectsAndKeys:
                        fileName, QTMovieFileNameAttribute,
                        async, QTMovieOpenAsyncOKAttribute,
                     nil];

movie = [QTMovie movieWithAttributes:attributes error:NULL];
```

Asynchronous Loading Summary

- QTMovieOpenAsyncOKAttribute will attempt to avoid blocking the initialization thread
  - YES by default
  - Work may or may not be done on a background thread
  - Will not result in failure to load if backgrounding not possible
- QTMovieOpenAsyncRequiredAttribute ***will*** avoid blocking the initialization thread
  - NO by default
  - Work ***will*** be done on a background thread if possible
  - May result in failure to load if backgrounding not possible

Usage Notes

A Taxonomy of Media Operations

Playback	**Editing**
Export	**Capture**

A Taxonomy of Media Operations

Playback

Editing

Export

Capture

A Taxonomy of Media Operations

Playback	Editing
Export	Capture

A Taxonomy of Media Operations

Playback	Editing
Export	**Capture**

QuickTime 7 and Media Operations

- Everything is richly represented as a Movie
- You say `NewMovieFrom*` or `-[QTMovie initWith*]`, and we prepare for
  - Playback
  - Editing
  - Export
- In some cases, a lot of work up front is required to build the Movie

QuickTime X

- You declare your intent
- We tailor what we do to match your intention
  - Hence, `QTMovieOpenForPlaybackAttribute`
  - If you say nothing, QTKit's default is to prepare for anything
  - As ever, fundamental operation is incremental, progressive, time-based

QTMovieLoadStateAttribute: An Encore

- `QTMovieLoadStateLoading`
  - Hold all queries
- `QTMovieLoadStateLoaded`
  - Attributes available
- `QTMovieLoadStatePlayable`
  - Media is available for playing
- `QTMovieLoadStatePlaythroughOK`
  - We project that media can be played without interruption
- `QTMovieLoadStateComplete`
- `QTMovieLoadStateError`

I Just Want to Play

- -autoplay monitors loading progress for you

```
// put a movie in a view and play it when ready

movie = [[QTMovie alloc] initWithAttributes:attributes
                                      error:NULL];
[movieView setMovie:movie];
[movie autoplay];
```

QTMovieLoadStateComplete:

What does it mean?

- We have a complete Movie representation
  - e.g. `-movieFormatRepresentation` can be called
- All media data is available on demand
  - e.g. `-writeToFile:attributes:` can be called with any options
    - Implies a particular media loading and caching model
- Is `QTMovieLoadStateComplete` needed in order to play?
  - In a word, no
  - When you set `QTMovieOpenForPlaybackAttribute`, the “complete” load state will not be signaled

Usage Restrictions When Opened for Playback

- Avoiding unreliable dependencies
- You're not informed which path we're taking
  - API restrictions when opened for playback
    - No editing
    - No saving
    - No archiving
    - No export
    - No access to QT7 primitives: Movie, Track, etc.
  - Restrictions pertain regardless of the path taken
  - NSException: `QTDisallowedForInitializationPurposeException`

Demo

Transitioning Your Application to QuickTime X

Transition Summary

- Declare your intent, win valuable prizes
  - `QTMovieOpenForPlaybackAttribute`
- Uncover usage that doesn't match your declared intent
  - Restrictions called out in Snow Leopard documentation, QTKit headers
  - Watch for `QTDisallowedForInitializationPurposeException`
  - Your choice when resolving usages that fall beyond the pale
    - Modify the behavior, e.g. use a different QTMovie
    - Retain the behavior, don't declare "open for playback"

Additional Benefits of QTMovieOpenForPlaybackAttribute

- 64-bit native, takes advantage of multiple cores, etc.
- Improved color fidelity
  - Details in Session 301
- Hardware codec, where available
  - Various recent models support H.264 decode in hardware
  - You get it if it isn't busy
    - Supports a limited number of simultaneous decoding sessions
    - Arbitrate your QTMovies via `-[QTMovie setActive:]`
- HTTP Live Streaming
  - New streaming technology
  - Details in Session 313

Multithreaded Applications

- All existing recommendations pertain
  - TN2125: Thread-safe Programming in QuickTime
- Initialize QTMovies on the main thread
  - Use `QTMovieOpenAsyncRequiredAttribute` to improve responsiveness
  - Migrate them if necessary to other threads

    Call `+[QTMovie enterQTKitOnThread]` on background threads before any other QTKit methods

    `-detachFromCurrentThread`/`-attachToCurrentThread`
    - Success depends on thread-safety of underlying decoders etc.
    - Have a fallback ready
- Only QTMovies attached to the main thread can play

Notes for Component Developers

I Want to Be Part of the Media System

- QuickTime 7
  - Image encoders/decoders
  - Audio encoders/decoders
  - Movie importers
  - Movie exporters
  - Pop quiz: how much of this was ready in QuickTime 1.0, c. 1991?

I Want to Be Part of the Media System

- QuickTime 7
  - Image encoders/decoders
  - Audio encoders/decoders
  - Movie importers
  - Movie exporters
  - Pop quiz: how much of this was ready in QuickTime 1.0, c. 1991?

I Want to Be Part of the Media System

- QuickTime X
  - Desire for extensibility
  - Requirement of security
  - Not ready for plug-in modules in Snow Leopard

QuickTime 7 Components

- `QTMovieOpenAsyncRequiredAttribute`
  - QTKit calls `NewMovieFromProperties` on a background thread
  - Thread-safety of your components will be exercised
- Thread-safe movie import components
  - See “TN2125: Thread-safe programming in QuickTime”
  - Attempt to display UI on a background thread can crash
  - Importers: To be thread-safe, media handlers, decoders must be thread-safe also
    - Sound Manager components are never thread-safe
    - If not safe, return `componentNotThreadSafeErr` (-2098)

Summary

Summary

- Use special initialization attributes to request new features
- Monitor movie load states during the loading process

```
QTMovieOpenForPlaybackAttribute
QTMovieOpenAsyncRequiredAttribute
```

Related Sessions

Color Management for Snow Leopard	Marina Friday 10:30AM
HTTP Live Streaming Introduction	Mission Wednesday 10:30AM
Sophisticated Media Playback Through Open Web Standards	Mission Wednesday 3:30PM

Labs

QTKit Lab	Graphics Lab B Tuesday 3:30PM
HTTP Live Streaming Lab	Graphics Lab B Wednesday 3:30PM

More Information

Allan Schaffer
Graphics Technology Evangelist
aschaffer@apple.com

Documentation, Sample Code, and Other Resources

http://developer.apple.com/graphicsimaging/

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09